REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	5/1/98	Final Technical 2/1/97-1/31/98

4. TITLE AND SUBTITLE
Genetic Analysis of Vertbrate Circadian Rhythmicity

5. FUNDING NUMBERS
F49620-97-1-0048
2312/CS
61102F

6. AUTHOR(S)
Gregory M. Cahill, Ph.D.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Houston
4800 Calhoun Road
Houston TX 77204

AFRL-SR-BL-TR-98-
0538

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of
Scientific Research/NL
110 Duncan Ave
Bolling AFB, DC 20332

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

19980722 036

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The goal of this project was to develop technology needed to identify vertebrate circadian clock genes. An understanding of basic molecular mechanisms of biological clocks is important in designing treatments of the performance deficits, sleep disorders and other problems associated with jet lag, shift work, and organic circadian clock disorders in humans. A novel automated image analysis system was developed to measure circadian locomotor rhythms. This system was used to measure behavioral rhythms in zebrafish, a vertebrate organism that is useful for large scale mutagenesis screens. A screen for dominant mutations that alter the period of zebrafish locomotor rhythms was initiated, and 1500 animals with different mutagenized genomes were tested. The progeny of 40 putative mutants are being tested to determine whether they carry authentic clock mutations.

14. SUBJECT TERMS
Circadian rhythms, Genetics

15. NUMBER OF PAGES 21

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

Summary

The goal of this project was to identify vertebrate circadian clock genes through mutational analysis of zebrafish circadian rhythm. The original proposal was for three years, and the specific aims included: 1) development of an automated image analysis method for measurement of locomotor activity rhythms; 2) a screen of 5,000 mutagenized individuals for dominant mutations; 3) genetic mapping of confirmed clock mutations; 4) characterization of mutant phenotypes, and 5) determination of the effects of visual system mutants on circadian rhythms. In the one year that was funded by the AFOSR, we accomplished the first specific aim and initiated the second specific aim. Specific aims 3-5 will be pursued with funding from other sources.

Publications

Bégay,V., Falcón, J., Cahill, G.M., Klein, D.C. and Coon.S.L. 1998. Transcripts Encoding Two Melatonin Synthesis Enzymes in the Teleost Pineal Organ: Circadian Regulation in Pike and Zebrafish but not in Trout. *Endocrinology*: **139**:905-912

Klein, D.C., Coon, S.L., Roseboom, P.H., Weller, J.L., Bernard, M., Gastel, J.A., Zatz, M., Iuvone, P.M., Rodriguez, I.R., Begay, V., Falcon, J., Cahill, G.M., Cassone, V.M. and Baler, R. 1997. The melatonin rhythm-generating enzyme: Molecular regulation of serotonin N-acetyltransferase in the pineal gland. *Rec. Prog. Horm. Res.* **52**: 307-358

Hurd, M.W., DeBruyne, J., Straume, M. and Cahill, G.M. Circadian rhythms of locomotor activity in zebrafish. In press, *Physiology and Behavior.*

Manuscript submitted

Cahill, G.M., Hurd, M.W. and Batchelor, M.M. Circadian rhythmicity in the locomotor activity of larval zebrafish. Submitted, *Neuroreport..*

Published Abstract

Hurd, M.W., DeBruyne, J, and Cahill, G.M. 1997 Circadian rhythms of locomotor behavior in zebrafish. *Soc. Neurosci. Abstr.* 23:1322.

Progress on Specific Aims

Specific Aim 1: Development of an automated video image analysis for measurement of circadian behavioral rhythms in larval zebrafish. This aim has been completed and reported in a manuscript submitted to the journal *Neuroreport.* The methods and results are described below. This project entailed development of a computer macro program, entitled Swimming Macro, written in Analytical Language for Images, and run within the the Optimas image processing program. The code is included in the appendix.

Methods: Specific Aim 1

Animals: The zebrafish used in these studies were of an AB strain that was obtained from the University of Oregon. They were raised to 10 days of age in a 14:10 light:dark cycle at 28.5°C, and they were fed *Paramecium* through the day before activity recording began. During activity recording they were housed individually in 0.75 ml wells. They were not fed during the 130-hr recording period. All protocols were reviewed and approved by the University of Houston Institutional Animal Care and Use Committee.

Recording apparatus: Fish were housed individually in up to 100 cylindrical wells (0.75 cm diam, 1 cm deep) drilled in a 10 X 10 grid in a 1.2 cm thick specimen plate made of translucent white polyethylene. A diffuse axial illuminator (Edmund Scientific, Barrington, NJ) placed above the wells provided even, infrared (>700 nm) illumination and eliminated reflection from the water surface. The wells were backlighted by a placing a mirror beneath the specimen plate; this enhanced the contrast of the fish relative to background. Images were collected from a CCD camera with a 28 mm lens, automatic gain control and shading correction (Hamamatsu Photonics, Hamamatsu, Japan). The specimen plate, illumination system and camera head were enclosed in a refrigerated incubator maintained at 24 ± 0.1°C. Monochrome images (640 X 480 pixels, 8 bit resolution) were captured by a Flashpoint 128 video card (Integral Technologies, Indianapolis, IN) in a computer with a Windows operating system and either a Pentium 166 or Pentium Pro II 300 processor (Intel, Santa Clara, CA).

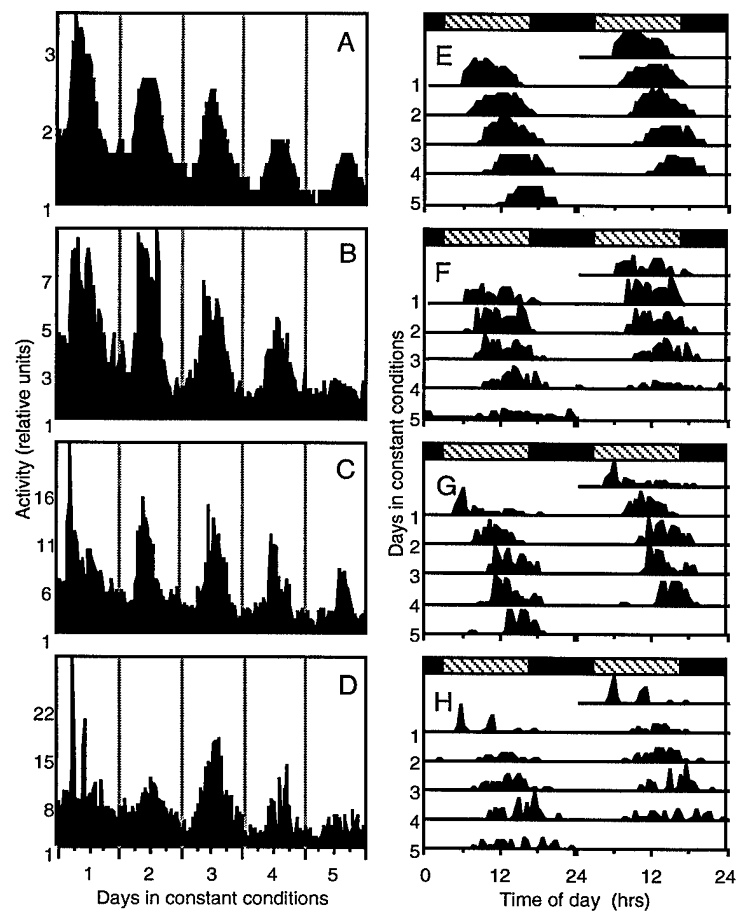
Data acquisition: Data were acquired automatically via Optimas (Seattle, WA) image analysis software, controlled by a macro written in Analytical Language for Images. The code fro this macro is inclluded in the appendix. Every 4 or 5 min, a series of 60 images was collected (1/sec) and stored in memory, then the distance that each fish moved in the series of images was measured. During set up of an experiment, the image frame was divided into rectangular cells enclosing each of the 100 wells, and a pixel value threshold that distinguished all of the fish from the lighter background was set. The coordinates of the centroid of each fish were determined for each frame of the series, and from these data the path through which each fish moved in the series of images was plotted. The length of the path of each fish was measured and stored in a text file, and the images in memory were discarded before reinitiating the cycle. The analysis of a series of 60 images of 100 fish required 2.5-4 min of processing time, depending on the microprocessor and version of Optimas that were used.

Two types of errors in locating the coordinates of the fish were encountered with this system: 1) In some cases, a fish was not recognized because too few pixels crossed threshold. When no fish was recognized, the previous location of the fish was recorded (i.e. no movement). This occurred most often when fish were inactive near the bottom of the well, so this probably did not contribute much to noise in the records. 2) In other cases, more than one object was recognized in a well, either because the fish image crossed threshold in two or more separate locations, or because a background area crossed threshold. In these cases the mean of the coordinates of all object centroids was recorded.

Experimental Design: Typically, behavioral recording began at the normal time of the light to dark transition on the tenth day post-fertilization and was continued for five to six days in constant infrared illumination. In another series of experiments, two groups of animals were exposed to opposite light cycles from five to ten days of age, and then they were monitored simultaneously in constant conditions to determine whether the recorded rhythms reflected the phase of the prior entraining light cycle.

Data analysis: Data were plotted and analyzed with Chrono II software (T. Roenneberg, University of Munich). The path lengths recorded during each 30 min period of an experiment were averaged, then plotted as a function of time. For actogram plots and periodogram analysis, a 26 h centered moving average was subtracted from the data to remove long term trends in overall activity levels. This results in positive values at times of peak activity, and negative values during troughs. In actograms, the positive values are plotted as a percent of the 26 hour mean, and negative values are omitted. Estimates of circadian period were determined by chi-square periodogram.

Figure 1. Circadian activity rhythms of larval zebrafish in constant conditions. A-D, Continuous records of activity. E-H, double-plotted actograms derived from the activity records in A-D. A and E, averaged activity of 100 fish during a five day experiment. B-D and F-H are records from 3 representative individuals to illustrate the range of activity patterns observed in that experiment. The light cycle prior to the experiment is indicated by the shaded (light) and black (dark) bars above the actograms. The actograms, which plot activity levels above the daily mean, emphasize that the fish were most active during the subjective day, and that the freerunning periods of the activity rhythms are longer than 24 hr.



Results: Specific Aim 1.

The locomotor activity of larval zebrafish in constant conditions was rhythmic with an average freerunning circadian period of 25.6 h. Examples of raw activity data and actograms for the average of a group of 100 fish and for 3 individuals are shown in Fig 1. Periodogram analysis indicated that over 95% of the animals expressed significant periodicity in the range of 23-28 hours in each of 5 experiments. The activity of every rhythmic animal was highest during the subjective day, with an approximately twofold difference between average trough and peak activity levels. A decrease in both trough and peak activity levels was observed over the course of the five day recording period in every experiment. Among experiments, average freerunning periods ranged from 25.2 to 26.1 hrs, and the within-experiment standard deviations in period ranged from 0.5 to 1.0 hours.

In order to ensure that the measured activity rhythms resulted from endogenous circadian rhythmicity, rather than some uncontrolled rhythmic variable in the environment or recording system, sibling groups of fish were raised in oppositely phased light:dark cycles, then activity of the two groups was monitored simultaneously in constant conditions. As shown in Fig. 2, the activity rhythms of the two groups of animals in constant conditions had opposite phases, with activity peaks occurring during the expected light period for each group. This demonstrates that the measured rhythms are driven by endogenous, entrainable circadian clocks, and can not be explained by extrinsic factors.

All zebrafish tested under these conditions survived the five day recording period without food or water changes. Some were removed from the wells at the end of the recording period and raised. All of these developed normally and began to breed at two to three months of age.

Conclusion: Specific Aim 1

The locomotor activity of larval zebrafish is regulated by a light-entrainable circadian clock. The resulting activity rhythms in constant conditions are robust, with consistent freerunning periods and a consistent phase relative to the prior light cycle. Activity rhythms of 100 individuals can be monitored simultaneously by automated video image analysis. This system should provide the efficiency and precision necessary for identification and characterization of zebrafish circadian clock mutants.

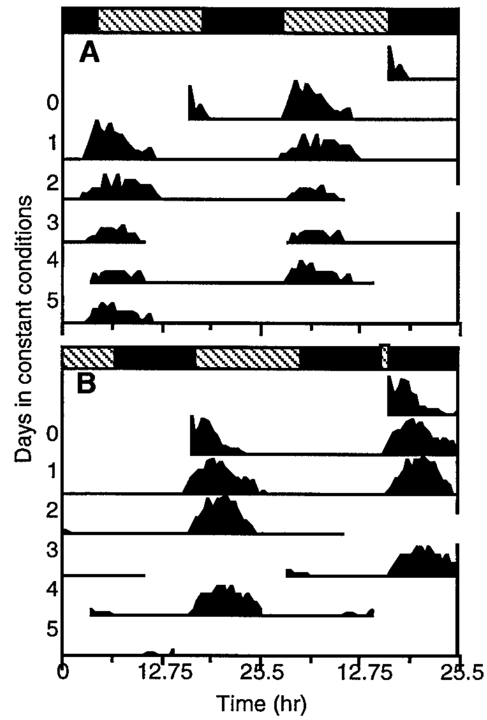


Figure 2. The phase of the zebrafish activity rhythm in constant conditions is determined by the preceding light:dark cycle. Double-plotted actograms of averaged activity from animals previously exposed to either a normal (A) or reversed (B) light:dark cycle, then monitored simultaneously (30 per group). The actograms are plotted on a 25.5 hr time scale, which approximates the freerunning periods of the rhythms. The light exposure during the 51 hours prior to placing the animals in constant conditions is indicated by the shaded bars above the graphs. The blank interval on Day 3 resulted from an 18 hr interruption of data collection.

Specific Aim 2: Mutagenesis and screen for dominant mutations in zebrafish circadian clock genes. We have made significant progress on this aim. Behavioral rhythms from approximately 1,500 mutagenized have been recorded and analyzed. Forty putative mutants have been selected on the basis of lengthened or shortened period of the circadian behavioral rhythm. The progeny of these 40 putative mutants are being tested for genetic transmission of clock mutations.

Methods, Specific Aim 2

Mutagenesis: Male zebrafish, 3-5 months old, were immersed 3-5 times for one hour in 3 mM N-ethyl-N-nitrosourea (ENU) to mutagenize progonial sperm cells. After one month recovery, these animals were crossed with normal females. Each of the resulting progeny is expected to be heterozygous for a unique combination of mutant loci.

Screening: Progeny of mutagenized males (F1s) were tested for changes in behavioral circadian rhythms by the method described under Specific Aim 1. Behavioral rhythms were measured for 5 days, and putative mutant individuals with freerunning periods shorter than 24 hours or longer than 27 hours were selected and raised to breeding age. Progeny of these animals (F2s, 15-30 each) are then tested to determine whether the period defect segregated with a 1:1 ratio.

Results, Specific Aim 2

To date, we have screened 1500 F1s. Forty of these animals that expressed circadian rhythms with long or short periods have been selected. The progeny of these mutants will be tested to determine whether the period changes are due to dominant mutations.

Conclusions, Specific Aim 2

The data to date indicate that it will be possible to identify circadian clock genes by mutational analysis in zebrafish. This component of the project is continuing with funding from another source.

Appendix:

Computer code for swimming macro.

```
/* ----------------------------------------------------------------------------------------------------
Title
        Swimming.Mac

By
        Matthew M. Batchelor
        Meyer Instruments, Inc.
        Houston, TX

Support Software
        Minimize.Mac
        Optdlg.Oml

Modification Log
        08-07-97: Originally based on TstDlg.Mac - MMB

---------------------------------------------------------------------------------------------------- */

// First, prevent recursive creation of this dialog
If (IsObject ("Swimming_hWndDlg")) {
        {
        Beep( );
        If ( ! Prompt ("Recursive attempt to create dialog box detected!\r\n\r\nContinue?", 2 ))
                Pause( );
        }
}

ChangeCursor (1);       // Make cursor into hourglass

// ------------------ VARIABLES used in the dialog box ----------------
CHAR
        Swimming_Directory = MacroPathAndName[0, ]; // Save current directory
INTEGER
        Swimming_hWndDlg; // The window handle of the new dialog box
BOOLEAN
        Swimming_IconOPTIMAS = FALSE; // Set TRUE to make OPTIMAS iconic

CHAR
        Swimming_T109 = ",
        Swimming_T110 = ",
        Swimming_T118 = ",
        Swimming_T141 = ",
        Swimming_T142 = ",
        Swimming_T143 = ",
        Swimming_T144 = "1",
        Swimming_T145 = "99",
        Swimming_T146 = "3",
        Swimming_T147 = "3",
        Swimming_T148 = Swimming_T144,
        Swimming_T149 = Swimming_T145,
```

```
        Swimming_T161 = '',
        Swimming_T162 = '',
        Swimming_T163 = '',
        Swimming_T999 = '';

CHAR
        Swimming_ArithmeticOpsNames [ , ] = '';

REAL
        Swimming_ROIWide;
REAL
        Swimming_ROIHigh;

INTEGER
        Swimming_NRows;
INTEGER
        Swimming_NCols;

REAL
        Swimming_SampleWide;
REAL
        Swimming_SampleHigh;

REAL
        Swimming_ROIImage [ , ];
REAL
        Swimming_ROIAnalyze [ , ];

Swimming_ROIImage = ROI ;
Swimming_ROIAnalyze = ROI;

REAL
        Swimming_DisplaySuperMArCentroids [ , , ] = '';

//If ( !IsWindow ( "Swimming_DisplaySuperMArCentroids" ) )
//      ViewBox ( Swimming_DisplaySuperMArCentroids , 0x0800 );

INTEGER
        Swimming_DataNoOfBlobs [ 10, 10, 60] = '';

REAL
        Swimming_DataCentroid [ 10, 10, 60, 2] = '';

REAL
        Swimming_DataLength [ 10, 10] = '';

INTEGER
        Swimming_Row ,
        Swimming_Col ,
        Swimming_Img ;

CHAR
        Swimming_cFileName [ , ] ;
INTEGER
```

```
        Swimming_cFileHandle ;

// ------------------- Startup code ------------------------------------

// Make OPTIMAS iconic ?
if (Swimming_IconOPTimas)
 RunMacro (PathVariable : "macros/minimize.mac");

// Load the DLL that handles custom dialog boxes
Swimming_hLib = LoadMacroLibrary( "optdlg.oml");

// Register function from windows which gives system tick count in msec
// Swimming_hGTC = Register ( "User" , "GetTickCount" , "%D" ) ;
// Swimming_hGTC = Register ( "Kernel32" , "GetTickCount" , "%D" ) ;

// --------------- FUNCTIONS used in the dialog box ------------------------

// Function to update information in the dialog box
Define Swimming_Update ( )
        {
        // Update ROI text
        Swimming_ROIWide = Swimming_ROIAnalyze [ 1 , 0 ] - Swimming_ROIAnalyze [ 0 , 0
] ;
        Swimming_T109 =
                ToText ( Swimming_ROIWide , "%.2f") :
                " (" : GetOrSetField ( ActiveCalibration , 103 ) : ".)"
                ;
        Swimming_ROIHigh = Swimming_ROIAnalyze [ 0 , 1 ] - Swimming_ROIAnalyze [ 1 , 1
] ;
        Swimming_T110 =
                ToText ( Swimming_ROIHigh , "%.2f") :
                " (" : GetOrSetField ( ActiveCalibration , 103 ) : ".)"
                ;

        // Update row and column variables
        FromText ( Swimming_T146 , Swimming_NRows );
        FromText ( Swimming_T147 , Swimming_NCols );
        Swimming_SampleWide = Swimming_ROIWide / (Real) Swimming_NCols ;
        Swimming_SampleHigh = Swimming_ROIHigh / (Real) Swimming_NRows ;
        }

// The init macro can do any special processing such as creating viewboxes
Define Swimming_InitMacro ( )
        {
        // Call update function to start everything off right
        Swimming_Update ( );
        }

// Button to prompt for path, prefix and extension
Define Swimming_B119 ( )
        {
        // Declare local variables
        Local Char TempPath [ ] ;
        Local Char TempFileName [ , ] ;
```

```
// Initialize TempPath
TempPath =
        FileSplitPath ( ImageFile ) [ 0 , ] ;
TempPath :=
        FileSplitPath ( ImageFile ) [ 1 , ] ;

// Prompt the user for the filename
TempFileName = GetFileName ( TempPath : "*.*" , -2 , -1 , TRUE ) ;

// Get file name (with error trap)
If ( TempFileName != FALSE )
        {
        // Open the image indicated by the user
        OpenImage ( TempFileName );

        // Remember the roi used by the image
        Swimming_ROIImage = ROI ;

        // Update the dialog box text
        Swimming_T141 = FileSplitPath ( TempFileName ) [ 0 , ] ;
        Swimming_T141 := FileSplitPath ( TempFileName ) [ 1 , ] ;
        Swimming_T142 = FileSplitPath ( TempFileName ) [ 2 , ] ;
        Swimming_T142 = Swimming_T142 [ 0 .. 3 ];
        Swimming_T143 = FileSplitPath ( TempFileName ) [ 3 , ] ;
        }
}

// Button to load images into memory
Define Swimming_B120 ( )
        {
        // Declare local variables
        Local Char
                TempFileName;
        Local Char
                TempCycleC;
        Local Char
                TempImageC;
        Local Integer
                Start,
                End;

        // Initialize local variables
        FromText ( Swimming_T144 , Start);
        FromText ( Swimming_T145 , End);

        // Clear names array
        Swimming_ArithmeticOpsNames = '';

        // Update user info
        Swimming_T118 = "Loading images into memory!";

        // Load all images from with the current prefix, cycle, and extension
        // into the arithmetic ops buffer in sequential order
```

```
	For ( j = Start ; j <= End ; j ++ )
		{
		For ( i = 0; i < 99; i ++ )
			{
			// -- Build the file name

			// Start with the path
			TempFileName = Swimming_T141;

			// Add the prefix
			TempFileName := Swimming_T142;

			// Add the Cycle
			If ( j < 1000 )
				TempCycleC = ToText ( j ) ;
			If ( j < 100 )
				TempCycleC = "0" : ToText ( j ) ;
			If ( j < 10 )
				TempCycleC = "00" : ToText ( j ) ;
			TempFileName := TempCycleC ;

			// Add the Image
			If ( i < 100 )
				TempImageC = ToText ( i ) ;
			If ( i < 10 )
				TempImageC = "0" : ToText ( i ) ;
			TempFileName := TempImageC ;

			// Add the extension
			TempFileName := Swimming_T143;

			// If the file exists
			If ( OpenFile ( TempFileName , 0x4000 ) != FALSE )
				{
				// Put it on the arithmetic ops stack
				FileToList ( TempFileName );
				// Remember the names on the stack
				Swimming_ArithmeticOpsNames ::=
					Swimming_T142 : TempCycleC : TempImageC :
Swimming_T143;
				}
			}
		// Update user info
		Swimming_T118 = "Images from Cycle " : ToText ( j ) : " loaded into memory!";
		}
	// Update user info
	Swimming_T118 = "Images loaded into memory!";
	}

// Button to playback images in memory
Define Swimming_B121 ( )
	{
	For ( i = 0 ; i < GetShape ( Swimming_ArithmeticOpsNames ) [ 0 ] ; i ++ )
		{
```

```
         StatusBar = "Arithmetic Ops Copy of " : Swimming_ArithmeticOpsNames [ i , ] ;
                SelectROI ( Swimming_ROIImage );
                ArithmeticOp ("Copy", Swimming_ArithmeticOpsNames [ i , ] );
                }
        }

// Button to clear images in memory
Define Swimming_B122 ( )
        {
        For ( i = 0 ; i < GetShape ( Swimming_ArithmeticOpsNames ) [ 0 ] ; i ++ )
                {
         StatusBar = "Deleting " : Swimming_ArithmeticOpsNames [ i , ] ;
                DeleteImage ( Swimming_ArithmeticOpsNames [ i , ] );
                }
        // Clear name array
        Swimming_ArithmeticOpsNames = '';
        }

// Button to set region of interest
Define Swimming_B123 ( )
        {
        // Prompt user
        If ( Prompt ( "Use current ROI as Analysis ROI?" , 2 ) )
                {
                // Remember this ROI
                Swimming_ROIAnalyze = ROI;
                }
                Else
                {
                // Activate ROI tool
                SelectROI ( );
                // Remember this ROI
                Swimming_ROIAnalyze = ROI;
                }

        // Call update function
        Swimming_Update ( );
        }

// Button to test sample layout
Define Swimming_B124 ( )
        {
        // Clear existing overlaid screen objects
        ClearScreen ( );

        SelectROI ( Swimming_ROIAnalyze );

        // Turn off screen updated
        BeginOrEndUpdateBlock( TRUE);

        // Loop through the rows
        For ( j = 0 ; j < Swimming_NRows ; j ++ )
                {
                // Loop through the columns
```

```
                For ( i = 0 ; i < Swimming_NCols ; i ++ )
                        {
                        // Create an area at the bounds of the ROIS
                        CreateArea
                                (
                                (
                                ( ROI [ 0 , 0 ] + ( (Real) i * Swimming_SampleWide ) ) :
                                ( ROI [ 0 , 1 ] - ( (Real) j * Swimming_SampleHigh ) )
                                )
                                ::
                                (
                                ( ROI [ 0 , 0 ] + ( (Real) i * Swimming_SampleWide ) ) :
                                ( ROI [ 0 , 1 ] - ( (Real) ( j + 1 ) * Swimming_SampleHigh ) )
                                )
                                ::
                                (
                                ( ROI [ 0 , 0 ] + ( (Real) ( i + 1 ) * Swimming_SampleWide ) ) :
                                ( ROI [ 0 , 1 ] - ( (Real) ( j + 1 ) * Swimming_SampleHigh ) )
                                )
                                ::
                                (
                                ( ROI [ 0 , 0 ] + ( (Real) ( i + 1 ) * Swimming_SampleWide ) ) :
                                ( ROI [ 0 , 1 ] - ( (Real) j * Swimming_SampleHigh ) )
                                )
                                );
                        }
                }
        // Turn on updates and refresh everything
        BeginOrEndUpdateBlock( FALSE);
        }

// Function to loop through the sample areas
Define Swimming_SampleLoop ( )
        {
        // Clear existing overlaid screen objects
//      ClearScreen ( );

        // Loop through the rows
        For ( j = 0 ; j < Swimming_NRows ; j ++ )
                {
                // Loop through the columns
                For ( i = 0 ; i < Swimming_NCols ; i ++ )
                        {
                        // Create an ROI
                        SelectROI
                                (
                                (
                                ( Swimming_ROIAnalyze [ 0 , 0 ] + ( (Real) i *
Swimming_SampleWide ) ) :
                                ( Swimming_ROIAnalyze [ 0 , 1 ] - ( (Real) j *
Swimming_SampleHigh ) )
                                )
                                ::
                                (
```

```
                                        ( Swimming_ROIAnalyze [ 0 , 0 ] + ( (Real) ( i + 1 ) *
Swimming_SampleWide ) ) :
                                        ( Swimming_ROIAnalyze [ 0 , 1 ] - ( (Real) ( j + 1 ) *
Swimming_SampleHigh ) )
                                        )
                                        );

                              // Set global variables equal to loop counters
                              Swimming_Row = j ;
                              Swimming_Col = i ;

                              // Run the macro to process this ROI
                              RunMacro ( Swimming_Directory : "Process/Process.Mac" );
                              }
                    }
          }

// Go Button to process images from disk
Define Swimming_B140 ( )
          {
          // Declare local variables
          LOCAL INTEGER
                    CycleStart, CycleEnd;

          // Initialize local variables
          FromText ( Swimming_T148 , CycleStart );
          FromText ( Swimming_T149 , CycleEnd );

          // Set up data extraction
          SetExport (Null, -2, FALSE);
          SetExport (mArCentroid, 1, TRUE);
          SetExport (ArCentroid, 1, TRUE);
          SetExport (mLnLength, 1, TRUE);
          SetExport (LnLength, 1, TRUE);

          // Initialize summary data vectors
          GLOBAL INTEGER
                    Swimming_DataNoOfBlobs [ 10, 10, 60] = "";

          GLOBAL REAL
                    Swimming_DataCentroid [ 10, 10, 60, 2] = "";

          GLOBAL REAL
                    Swimming_DataLength [ 10, 10] = "";

          // Loop through the requested cycles
          For ( CycleI = CycleStart ; CycleI <= CycleEnd ; CycleI ++ )
                    {
                    // Reset the from and through numbers in the images from disk
                    Swimming_T144 = ToText ( CycleI );
                    Swimming_T145 = ToText ( CycleI );

                    // Clear overlaid screen objects
                    ClearScreen ( );
```

```
// Call the function which loads the images into memory
Swimming_B120 ( );

// Play back the stack in memory
For ( ii = 0 ; ii < GetShape ( Swimming_ArithmeticOpsNames ) [ 0 ] ; ii ++ )
        {
        // Update user info
        Swimming_T118 =
                "Processing Image Number " :
                ToText ( ii ) :
                " of " :
                ToText ( GetShape ( Swimming_ArithmeticOpsNames ) [ 0 ] ) :
                "!";

  StatusBar = "Arithmetic Ops Copy of " : Swimming_ArithmeticOpsNames [ ii , ] ;
        SelectROI ( Swimming_ROIImage );
        ArithmeticOp ("Copy", Swimming_ArithmeticOpsNames [ ii , ] );

        BeginOrEndUpdateBlock ( TRUE );

        SelectROI ( Swimming_ROIAnalyze );
        Swimming_Img = ii;
        Swimming_SampleLoop ( );

        BeginOrEndUpdateBlock ( FALSE );

        }

// Loop through the rows
For ( j = 0 ; j < Swimming_NRows ; j ++ )
        {
        // Loop through the columns
        For ( i = 0 ; i < Swimming_NCols ; i ++ )
                {
                MultipleMode = FALSE;
                Local Real My_Points = Swimming_DataCentroid [ i , j , 0 ..
Swimming_Img  , 0 .. 2 ];
                Temp_IDLine = CreateLine ( My_Points );
                If ( Temp_IDLine != FALSE )
                        {
                        Extract ( );
                        Swimming_DataLength [ i, j] = LnLength;
                        }
                        Else
                        {
                        Swimming_DataLength [ i, j] = 0.;
                        }
                MultipleMode = TRUE;
                }
        }

// If user opened data file send data to file
If ( Swimming_cFileName )
```

```
{
// Open data file
Swimming_cFileHandle = OpenFile ( Swimming_cFileName, 0x0001);
// Move to end of file
PositionFile (Swimming_cFileHandle , 0L, 2);
// Send data to data file

// -- Column Headings
If ( CycleI == CycleStart )
        {
        Swimming_cOutLine =
                "Col\t" ;
        // Loop through the rows
        For ( j = 0 ; j < Swimming_NRows ; j ++ )
                {
                // Loop through the columns
                For ( i = 0 ; i < Swimming_NCols ; i ++ )
                        {
                        Swimming_cOutLine :=
                                ToText ( i ) : "\t" ;
                        }
                }
        Swimming_cOutLine :=
                "\r\n" ;
        WriteFile ( Swimming_cFileHandle, Swimming_cOutLine );
        }

// -- Row Headings
If ( CycleI == CycleStart )
        {
        Swimming_cOutLine =
                "Row\t" ;
        // Loop through the rows
        For ( j = 0 ; j < Swimming_NRows ; j ++ )
                {
                // Loop through the columns
                For ( i = 0 ; i < Swimming_NCols ; i ++ )
                        {
                        Swimming_cOutLine :=
                                ToText ( j ) : "\t" ;
                        }
                }
        Swimming_cOutLine :=
                "\r\n" ;
        WriteFile ( Swimming_cFileHandle, Swimming_cOutLine );
        }

// -- Length Data
Swimming_cOutLine =
        Swimming_ArithmeticOpsNames [ Swimming_Img , ] : "\t";
// Loop through the rows
For ( j = 0 ; j < Swimming_NRows ; j ++ )
        {
        // Loop through the columns
```

```
                    For ( i = 0 ; i < Swimming_NCols ; i ++ )
                        {
                        Swimming_cOutLine :=
                            ToText ( Swimming_DataLength [ i , j ] ) : "\t" ;
                        }
                    }
                Swimming_cOutLine :=
                    "\r\n" ;
                WriteFile ( Swimming_cFileHandle, Swimming_cOutLine );


                // Close data file
                CloseFile ( Swimming_cFileHandle );
                }

            // Call the function which clears the images in memory
            Swimming_T118 = "Deleting Images from Stack!";
            Swimming_B122 ( );

            }

        // Update user info
        Swimming_T118 = "Finished Processing Images!";

        }

// Button to prompt for data file info
Define Swimming_B157 ( )
        {
        // Declare local variables
        Local Char TempPath [ ] ;

        // Initialize TempPath
        TempPath =
            FileSplitPath ( ImageFile ) [ 0 , ] ;
        TempPath :=
            FileSplitPath ( ImageFile ) [ 1 , ] ;

        // Prompt the user for the filename
        TempFileName = GetFileName ( TempPath : "*.*" , TRUE , 1, FALSE ) ;

        // Get file name (with error trap)
        If ( TempFileName != FALSE )
            {
            // Update the dialog box text
            Swimming_T161 = FileSplitPath ( TempFileName ) [ 0 , ] ;
            Swimming_T161 := FileSplitPath ( TempFileName ) [ 1 , ] ;
            Swimming_T162 = FileSplitPath ( TempFileName ) [ 2 , ] ;
            Swimming_T163 = FileSplitPath ( TempFileName ) [ 3 , ] ;

            Swimming_cFileName = TempFileName ;
            // Open data file
            Swimming_cFileHandle = OpenFile ( Swimming_cFileName, 0x0001);
            // Move to end of file
```

```
            PositionFile (Swimming_cFileHandle , 0L, 2);
            // Send data to data file
            // Close data file
            CloseFile ( Swimming_cFileHandle );
            }
    }

Define Swimming1_TerminateMacro ( )
    {
    // Unload the dialog box library
    LoadMacroLibrary(Swimming_hLib);

    // If OPTIMAS was iconic, bring it back
    If (Swimming_IconOPTIMAS)
            RunMacro (PathVariable : "macros/maximize.mac");

    // Wildcard delete of "Swimming_.*"
    ObjectWildCardList ("Swimming_.*", 2);

    BeginOrEndUpdateBlock ( FALSE );
    }

// ------------------ LINK OBJECT creation --------------------------
OBJECT_ID Swimming_Links[,];
Swimming_Links =
// Text
109 : ObjectID ( Swimming_T109 ) : 0 : 0 : 0 ::
110 : ObjectID ( Swimming_T110 ) : 0 : 0 : 0 ::
118 : ObjectID ( Swimming_T118 ) : 0 : 0 : 0 ::
// Buttons
119 : 0 : ObjectID ( Swimming_B119 ) : 0 : 0 ::
120 : 0 : ObjectID ( Swimming_B120 ) : 0 : 0 ::
121 : 0 : ObjectID ( Swimming_B121 ) : 0 : 0 ::
122 : 0 : ObjectID ( Swimming_B122 ) : 0 : 0 ::
123 : 0 : ObjectID ( Swimming_B123 ) : 0 : 0 ::
124 : 0 : ObjectID ( Swimming_B124 ) : 0 : 0 ::
140 : 0 : ObjectID ( Swimming_B140 ) : 0 : 0 ::
157 : 0 : ObjectID ( Swimming_B157 ) : 0 : 0 ::
// Edit Boxes
141 : ObjectID ( Swimming_T141 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T141 ) : 0 ::
142 : ObjectID ( Swimming_T142 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T142 ) : 0 ::
143 : ObjectID ( Swimming_T143 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T143 ) : 0 ::
144 : ObjectID ( Swimming_T144 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T144 ) : 0 ::
145 : ObjectID ( Swimming_T145 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T145 ) : 0 ::
146 : ObjectID ( Swimming_T146 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T146 ) : 0 ::
147 : ObjectID ( Swimming_T147 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T147 ) : 0 ::
```

```
148 : ObjectID ( Swimming_T148 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T148 ) : 0 ::
149 : ObjectID ( Swimming_T149 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T149 ) : 0 ::
161 : ObjectID ( Swimming_T161 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T161 ) : 0 ::
162 : ObjectID ( Swimming_T162 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T162 ) : 0 ::
163 : ObjectID ( Swimming_T163 ) : ObjectID ( Swimming_Update ) : ObjectID (
Swimming_T163 ) : 0 ::
// Dummy
999 : 0 : 0 : 0 : 0 ;

// ------------------ Display the dialog box!!! ------------------------
OPTCreateDialog (
        Swimming_Directory : "Swimming.dlg", // Dialog template file, replace this!
        Swimming_Links, // Control Link
        Swimming_InitMacro, // Init Macro
        Swimming1_TerminateMacro, // Close Macro (note unique prefix!)
        hWndVideo, // Parent window handle
        Swimming_hWndDlg, // handle to the new dialog
        , // Title bar object
         // hMenu
        );

ChangeCursor (0);      // Make cursor into arrow
```